Department Of
Software Engineering

Delhi Technological University

2011-2015

Flight Management Unit

Submitted in the partial fulfillment of the
requirements for Minor Project in B.Tech. 6th Semester

Authors:
Asheesh Ranjan 2K11/SE/012
Pranav Jetley 2K11/SE/049
Varun Kalra 2K11/SE/081
Mentor:
Mr. Manoj Kumar

Department of Computer Science

Delhi Technological University

Certificate

This is to certify that this is a bonafide record of the project presented by the students whose names are given below in partial fulfilment of the requirements of Minor Project 2 for the degree of Bachelor of Technology in Software Engineering.

Roll No	Names of Students
Asheesh Ranjan	2K11/SE/012
Pranav Jetley	2K11/SE/049
Varun Kalra	2K11/SE/081

Manoj Kumar
Assistant Professor
Department of Computer Science
Delhi Technological University

Date: May 1, 2014

Acknowledgements

We would like to express our sincere gratitude and thank our project guide Mr. Manoj Kumar for giving the opportunity to work on this topic. We are indebted to him for his generosity, expertise and guidance, which we have received while working on this project. Besides our guide, we would like to thank entire teaching and non-teaching staff in the department for all their support and motivation during this work.

Abstract

The project focusses on developing an unmanned aerial vehicle with an onboard flight management unit for controlling the flight as per the remote operator's commands and a customized imagery system capable of capturing & transmitting live video with simultaneous processing done on it to deliver actionable data. The design discussed in this report is based on the development of UAV quad rotor helicopter (quadcopter) with emphasis on the flight management unit and the control system.

The quadcopter consists of four brushless outrunner DC motors, four electronic speed controllers, an Arduino Uno microcontroller board, Avionic RCB7X receiver and lithium polymer batteries, a Raspberry Pi, charged couple camera, receiver and a transmitter and lipo batteries for charging the Pi. The remote operator will be using a Avionic RCB7X controller. The software will be running onboard the quadcopter and will have sensor inputs from multiple sensors. We will be using a 9 degrees of freedom inertial measurement unit and an ultrasonic distance sensor. Given availability of sufficient funds in the future, we will incorporate other range sensors for more stable flight. The output will be the voltage control to the four electronic speed controllers.The receiver and transmitter will be integrated with the raspberry pi for transferring of live video feed to a laptop at the ground station. The flight management unit will be designed using concepts of control theory, signal smoothing, and general data structures and algorithms. On the other hand the image processing unit will be designed using OpenCV library, utilising various algorithms for image compression, image smoothing, image segmentation and object detection.

In the future, we intend to replace the remote operation of the quadcopter with on-board autonomous flight control. This will require addition of other sensors and better image processing hardware.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An unmanned aerial vehicle (UAV), commonly known as drone, is an aircraft without a human pilot aboard. Its flight is controlled either autonomously by onboard computers or by the remote control of a pilot on the ground or in another vehicle. The typical launch and recovery method of an unmanned aircraft is by the function of an automatic system or an external operator on the ground. There are a wide variety of UAV shapes, sizes, configurations, and characteristics. Most recently, artificial intelligence is being incorporated into UAVs. Autonomous take off, landing, and flying capability is being incorporated into the current generation of UAVs. Autonomy is moving from supervised systems to fully autonomous. Ongoing research focusses on path planning, decision making, obstacle avoidance and other AI based techniques for UAVs. However, fully autonomous UAVs are restricted to military operations at present.

UAVs usually consist of an flight management unit (FMU). An FMU is a low latency autopilot suitable for fixed wing, multi rotors, helicopters, cars, boats and any other robotic platform that can move. These are usually used on research and amateur projects. Full fledged commercial quadrotos usually utilise an flight management module or flight management system. An FMU is an expandable, modular system comprising the Flight Management Unit (autopilot) and a number of optional interface modules. In addition to the versatility of the hardware platform, FMU's have a sophisticated, modular, realtime software environment. Most FMUs have support for new sensors, peripherals and expansion modules due to standardized interface protocols between software components.

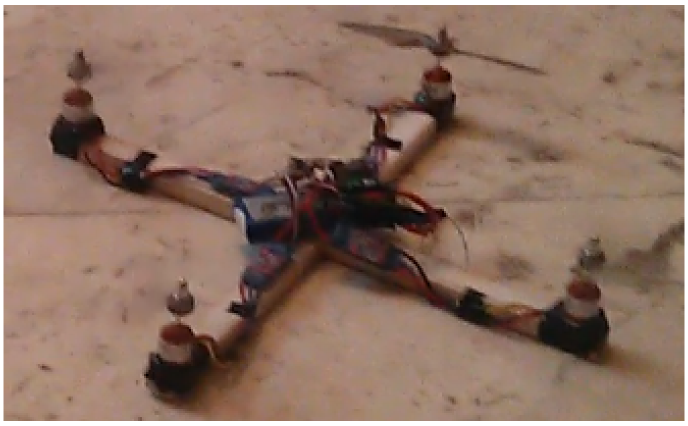


Figure 1.1: Our micro aerial vehicle platform

1.1 Micro Aerial Vehicles

Since 2003, the robotics community has been showing a growing interest in Micro Aerial Vehicle (MAV) development. The scientific challenge in MAV design and control in cluttered environments and the lack of existing solutions was very motivating. On the other hand, the broad field of applications in both military and civilian markets was encouraging the funding of MAV related projects. At the same time, the Autonomous Systems Laboratory (ASL) had already accumulated a large experience on ground-based robots with excellent results. Several theses were conducted on localization, navigation, obstacle avoidance etc. The limitations of ground-based robots in rough terrain and the recent progress in micro technology pushed us towards developing new mobility concepts. This includes flying systems on which one could apply the techniques already developed on ground-based robots. However, the task is not trivial due to several open challenges. In the field of sensing technologies, industry can currently provide a new generation of integrated micro Inertial Measurement Unit (IMU) composed generally of Micro Electro-Mechanical Systems (MEMS) technology inertial and magneto-resistive sensors. The latest technology in high density power storage offers about 190 Wh/kg which is a real jump ahead especially for micro aerial robotics. This technology was originally developed for hand-held applications and is now widely used in aerial robotics. The cost and size reduction of such systems makes it very interesting for the civilian market. Simultaneously, this reduction of cost and size implies performance limitation and thus a more challenging control. Moreover, the miniaturization of inertial sensors imposes the use of MEMS technology, which is still much less accurate than the conventional sensors because of noise and drift. The use of low-cost IMUs is synonym of less efficient data processing and thus a bad orientation data prediction in addition to a weak drift rejection. On the other hand, and in spite of the latest progress in miniature actuators, the scaling laws are still unfavorable and one has to face the problem of actuator saturation. That is to say, even though the design of micro aerial robots is possible, the control is still a challenging goal. It was decided from the beginning of this thesis to work on a particular VTOL configuration: the quadrotor. The interest comes not only from its dynamics, which represent an attractive control problem, but also from the design issue. Integrating the sensors, actuators and intelligence into a lightweight vertically flying system with a decent operation time is not trivial.

MAVs are usually deployed for military and special operations like surveillance and reconnaissance of hostile territory, patrolling border areas to detect intrusions. They are being used for a growing number of civil applications, such as policing, fire fighting and non-military security work, such as surveillance of pipelines. As a tool for search and rescue, UAVs can help find humans lost in the wilderness, trapped in collapsed buildings, or adrift at sea. In India, they were recently used in successfully locating survivors during the 2013 Uttarakhand floods, and are used by the Border Security Force (BSF) for patrolling the Indo-Pak border in the arid regions Prime Air programme in which UAVs will be used for product deliveries. Another company utilizing UAVs is Dominos, which attempts to deliver pizzas using UAVS.

1.2 Control System

Control theory is an interdisciplinary branch of engineering and mathematics that deals with the behavior of dynamical systems with inputs. The external input of a system is called the reference. When one or more output variables of a system need to follow a certain reference over time, a controller manipulates the inputs to a system to obtain the desired effect on the output of the system. The objective of control theory is to calculate solutions for the proper corrective action from the controller that result in system stability, that is, the system will hold the set point and not oscillate around it.

There are predominantly two types of control systems, open loop and closed loop systems. An open-loop controller, also called a non-feedback controller, is a type of controller that computes its input into a system using only the current state and its model of the system. A characteristic of the open-loop controller is that it does not use feedback to determine if its output has achieved the desired goal of the input. Consequently, an open-loop system cannot correct any errors and it also may not compensate for disturbances in the system. A closed-loop controller, on the other hand, uses feedback to control states or outputs of a dynamical system.

Closed-loop controllers have various advantages over open-loop controllers such as disturbance rejection, guaranteed performance even with model uncertainties, when the model structure does not match perfectly the real process and the model parameters are not exact, unstable processes can be stabilized, reduced sensitivity to parameter variations, and improved reference tracking performance. A common closed-loop controller architecture is the PID controller.

1.2.1 LTI

Linear time-invariant (LTI) theory, comes from applied mathematics and has direct applications in signal processing, control theory, and other technical areas. It investigates the response of a linear and time-invariant system to an arbitrary input signal. Trajectories of these systems are commonly measured and tracked as they move through time but in applications like image processing and field theory, the LTI systems also have trajectories in spatial dimensions.

Linearity of a system refers to its property of additive superposition. This essentially means that if we are to excite a system with input signal x and get an output as X, and excite the same system with signal y to get an output Y, then the system will yield an output response of magnitude $X + Y$ when excited by the inputs x and y together. A system is said to be time invariant it obeys the following time-shift invariance property:

If the response to the input signal x(t) is

$$y(t) = S[x(t)] \tag{1.1}$$

then for any real constant k,

$$y(t-k) = S[x(t-k)] \tag{1.2}$$

This is an idealistic way of looking into a system, as no system designed has been found to be entirely linear or time invariant. We can only aim to attain these properties with as much perfection as possible but practically, we cannot reach there entirely. A LTI system is important from an analytical point of view as it helps us in realising the entire system mathematically.

1.2.2 PID

PID is an acronym for Proportional-Integral Derivative, referring to the three terms operating on the error signal to produce a control signal. If u(t) is the control signal sent to the system, y(t) is the measured output and r(t) is the desired output, and tracking error e(t)=r(t)- y(t), a PID controller has the general form

$$u(t) = K_P e(t) + K_I \int e(t)\mathrm{d}t + K_D \frac{\mathrm{d}}{\mathrm{d}t} e(t). \tag{1.3}$$

The desired closed loop dynamics is obtained by adjusting the three parameters KP, KI and KD, often iteratively by ”tuning” and without specific knowledge of a plant model. PID controllers are the most well established class of control systems: however, they cannot be used in several more complicated cases, especially if MIMO systems are considered.

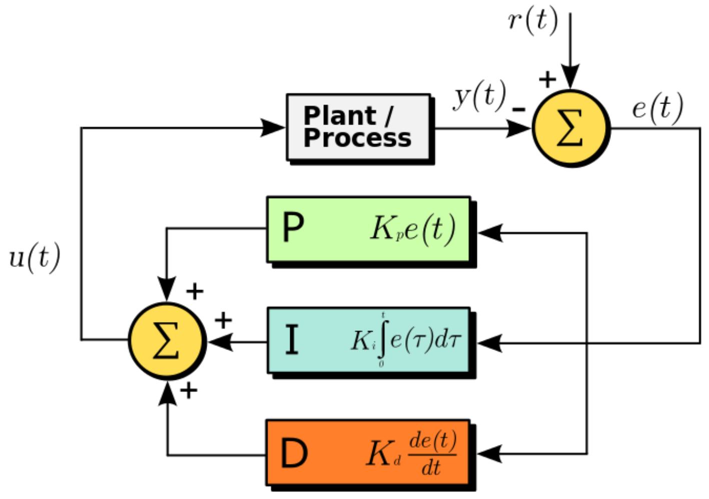


Figure 1.2: A proportional-integral-derivative controller

1.3 Digital Signal Processing

For remote operation, MAVs are operated by a radio controlled transmitter receiver system. The control signals are usually transmitted across a fixed frequency range. As they travel through the channel, these signals are subject to interference by noise, which can corrupt the control signal and compromise the functionality of the MAV. A variation in the control signal to the MAV will cause abrupt changes in motor RPM, leading to varying thrust values.

This variation in the thrust generated by the propellers can significantly change the orientation of the craft, in turn compromising flight stability. Thus, there is a need to filter out the noisy part of the signal to ensure that the MAV responds reliably to different control inputs. This field of DSP addresses the removal of noise in a control signal.

Digital signal processing and analog signal processing are subfields of signal processing. Digital signal processing is the mathematical manipulation of an information signal to modify or improve it in some way. It is characterized by the representation of discrete time, discrete frequency, or other discrete domain signals by a sequence of numbers or symbols and the processing of these signals. The goal of DSP is usually to measure, filter and/or compress continuous real-world analog signals. The first step is usually to convert the signal from an analog to a digital form, by sampling and then digitizing it using an analog-to-digital converter (ADC), which turns the analog signal into a stream of numbers. However, often, the required output signal is another analog output signal, which requires a digital-to-analog converter (DAC). Even if this process is more complex than analog processing and has a discrete value range, the application of computational power to digital signal processing allows for many advantages over analog processing in many applications, such as error detection and correction in transmission as well as data compression.

DSP applications include: audio and speech signal processing, sonar and radar signal processing, sensor array processing, spectral estimation, statistical signal processing, digital image processing, signal processing for communications, control of systems, biomedical signal processing, seismic data processing, etc. DSP algorithms have long been run on standard computers, as well as on specialized processors called digital signal processor and on purpose-built hardware such as application-specific integrated circuit (ASICs). Today there are additional technologies used for digital signal processing including more powerful general purpose microprocessors, field-programmable gate arrays (FPGAs), digital signal controllers (mostly for industrial apps such as motor control), and stream processors, among others.

1.4 Control System Tuning

Tuning a control loop is the adjustment of its control parameters to the optimum values for the desired control response. Stability (no unbounded oscillation) is a basic requirement, but beyond that, different systems have different behavior, different applications have different requirements, and requirements may conflict with one another.

PID tuning is a difficult problem, even though there are only three parameters and in principle is simple to describe, because it must satisfy complex criteria within the limitations of PID control. There are accordingly various methods for loop tuning including manual, automated, and certain patented techniques. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively inefficient, particularly if the loops have response times on the order of minutes or longer. The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and on the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Designing and tuning a PID controller appears to be conceptually intuitive, but can be hard in practice, if multiple (and often conflicting) objectives such as short transient and high stability are to be achieved. PID controllers often provide acceptable control using default tunings, but performance can generally be improved by careful tuning, and performance may be unacceptable with poor tuning. Usually, initial designs need to be adjusted repeatedly through computer simulations until the closed-loop system performs or compromises as desired. Some processes have a degree of nonlinearity and so parameters that work well at full-load conditions don't work when the process is starting up from no-load; this can be corrected by using different parameters in different operating regions.

1.5 Navigation

For any mobile device, the ability to navigate in its environment is important. Avoiding dangerous situations such as collisions and unsafe conditions (temperature, radiation, exposure to weather, etc.) comes first, but if the robot has a purpose that relates to specific places in the robot environment, it must find those places.

Robot navigation means the robot's ability to determine its own position in its frame of reference and then to plan a path towards some goal location. In order to navigate in its environment, the robot or any other mobility device requires representation, i.e. a map of the environment and the ability to interpret that representation. Navigation can be defined as the combination of the three fundamental competences self-localisation, path planning, map-building and map interpretation.

Map in this context denotes any one-to-one mapping of the world onto an internal representation. Robot localization denotes the robot's ability to establish its own position and orientation within the frame of reference. Path planning is effectively an extension of localisation, in that it requires the determination of the robot's current position and a position of a goal location, both within the same frame of reference or coordinates. Map building can be in the shape of a metric map or any notation describing locations in the robot frame of reference.

1.5.1 Visual Navigation

Vision-Based navigation uses optical sensors including laser-based range finder and photometric cameras using CCD arrays to extract the visual features required to for localization in the surrounding environment. However, there are a range of techniques for navigation and localization using vision information, the main components of each technique are representations of the environment, sensing models and localization algorithms. These may further be classified under indoor navigation and outdoor navigation techniques.

1.5.2 Image Processing

Image processing is any form of signal processing for which the input is an image, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. Image processing usually refers to digital image processing, but optical and analog image processing also are possible. The acquisition of images (producing the input image in the first place) is referred to as imaging. The two types of methods used for Image Processing are Analog and Digital Image Processing. Analog or visual techniques of image processing can be used for the hard copies like printouts and photographs. Image analysts use various fundamentals of interpretation while using these visual techniques. The image processing is not just confined to area that has to be studied but on knowledge of analyst. Association is another important tool in image processing through visual techniques. So analysts apply a combination of personal knowledge and collateral data to image processing. Digital Processing techniques help in manipulation of the digital images by using computers. As raw data from imaging sensors from satellite platform contains deficiencies. To get over such flaws and to get originality of information, it has to undergo various phases of processing. Digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. The image processing process generally consists of the following three steps.

1. Import the image with the help of an optical scanner or digital photography
2. Analyze and manipulate the image that included performing compression, image enhancement, spotting patterns that are not visible to the human eye
3. The third stage is the output which basically consists of the manipulated image or analysis report.

1.5.3 Computer Vision

Computer vision is the science of endowing computers or other machines with vision, or the ability to see. Basically it is giving machines or computers the power to see and visualize things in the same way as humans do.Humans make use of their brain and their eyes in order to gain a visual perspective of their environment. The field of computer vision attempts to give a comparable capability to a computer or other machine via the extraction, processing, and analysis of relevant information from an image, sequence of images, or multi-dimensional data.

Chapter 2

Literature Survey

2.1 MAV

In order to understand the quad-rotor platform better, we went through [4]. This paper presented an overview of the application potential and design challenges of MAVs, defined as small enough to be practical for a single-person transport and use. This paper also looked into different types of MAVs such as fixed-wing, rotary- wing, ornithopters (bird-like flapping) and entomopters (insect-like flapping). Thus, we were able to decide a size that would classify our flying robot as a MAV. We were also able to understand the scope of the functionality which the MAV would provide.

Next, we went through case studies of various other university teams who had embarked on a similar project. In [8], we learnt about flight mechanics, and the governing torque and force relations. This paper provided the basic dynamic model of the quad-rotor and formed the basis for our further exploration in the subject matter.

To understand the various electronic flight components including controllers and sensors that would be required for building a functional quad-rotor we explored [12]. This paper provided a detailed list and description of components that would be required. It delved into the principals behind and functioning of each of the components. Thus, we were able to obtain a basic understanding of the hardware we would be needing for our project.

2.2 Control System

To implement a functional FMU, we delved into the field of control systems. [21] provided a basic understanding of an LTI system, a standard control system in the field of robotics. We were able to comprehend the complexities involved in the design of a responsive and stable control system. An integral part of a control system is a PID controller. [?, ?] provided us with a thorough understanding of various controllers that are used within LTI systems. We also explored various tuning strategies that would be helpful in resolving the constants that would be required for the optimal performance of a PID controller.

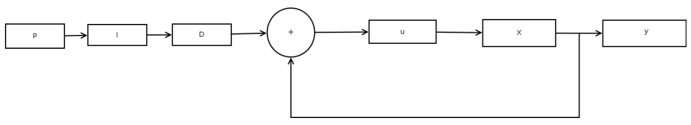


Figure 2.1: High level view of control systems

For the tuning of PID and control system constant parameters, [17] motivated us to develop our very own test bed, which allowed unrestricted motion on a single axis. We used the Ziegler-Nichols tuning method on the test bed to achieve the optimal values of the constant parameters. The Ziegler-Nichols method is a heuristic method which requires multiple iterations by repeatedly changing the constant values to obtain a desired level of performance [20].

As our platform was based on a microcontroller with very limited processing capabilities, we had to pay special attention to the refresh rate of the control signals that were going out to the ESCs. The ESCs typically require a very high refresh rate in the 80-120 Hz range for responsive and stable control. [16] motivated us to accurately estimate the time being taken in the execution of the control system and to attempt to micro-optimize sections of the implementation so as to increase the refresh rate.

2.3 DSP

One problem that we expected based on our previous experience was noise in the wireless signals that are used for control of mobile robots. The error introduced into a signal during transmission from the sender to the receiver is called noise. The presence of noise in a signal impairs the signal to noise ratio, which is a measure of signal quality, and causes fluctuation in the signal value. This in turn, makes the signal inaccurate and unreliable and may even lead to a loss of data. Noise can either be generated at the transmitter side circuit, in the transmission channel, or at the receiver side circuit. The types of noise being generated can be categorized depending on the cause. Random fluctuations in the current flowing in the circuits of the transmitter and receiver may be a source of shot noise. Flicker noise may occur in electronic devices and can show up as irregularities in the conducting wires causing fluctuation in the voltage and current values. Another type of noise is transient noise, which occurs during signal transmission and is caused by interference in the transmission channel. These are short pulses followed by decaying low frequency oscillations. Another source of noise is the thermal agitation of electrons inside a conductor, called thermal noise. Lastly, a major source of noise in our experiment is quantization noise. The analog signal sent by the transmitter is converted to a (PWM) signal before being sent to the motors. The rounding off of errors that occur during this process of signal conversion is known as quantization error, which presents itself as noise in the original signal.

Signal smoothing is a process that attempts to capture essential information while leaving out the noise in the signal, by interpolating the raw signal to estimate the original signal. As our system updates control signals in real time, we have considered filters in the time domain for signal smoothing[7] i.e. the simple moving average (SMA), cumulative moving average (CMA), exponential moving average (EMA)[6], the Savitzky-Golay (SG) filter, and the Ramer-Douglas-Peucker (RDP) algorithm[5]. These time domain filters usually have a finite window width and the window moves along the data set, as shown in the figure below. The data points in the smoothing window are assigned different weights and the actual value is interpolated. The weighting function is the primary factor differentiating the algorithms.

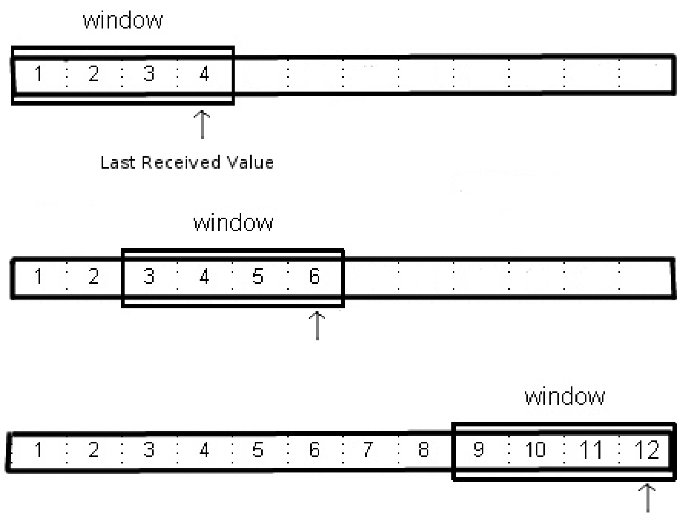


Figure 2.2: The window moves along the input sequence

2.4 Image Processing

In order to understand the full model of how an image processing system works in UAVs, we explored project reports of other teams who had worked on similar projects. [24],[23],[25] were studied in order to find out the image processing architecture. OpenCV, a library, was used for designing and implementing the image processing algorithms. None of us had worked on this library before. In order to learn OpenCV, [27] and documentation was studied.

2.4.1 Processing unit

In order to design the image processing unit, various components such as a camera, a processor to perform the necessary image manipulations, and a transmitter-receiver were needed. Raspberry Pi was chosen as the processor as it is small, credit card sized computer and has the best performance to cost ratio. RPI camera was used for taking the digital images as it could easily be plugged in the RPI and provided high quality pictures and videos. In order to understand how the RPI camera works, the RPI camera documentation was looked into. At the end we required a transmitter-receiver for transmitting live video feed from quad-copter to the remote station while the quad-copter is hovering. Various calculations were made so as to find out the requirements of the transmitter i.e. transfer data rate, cost, power consumption. We needed a transmitter that could provide a data rate of approx 1 MBps and range of more than 1 km and be cheap at the same time. Various options were looked into such as XBEE transmitter, Bluetooth but both of them had some problems. XBEE has a slow data rate where as the power consumption and price of bluetooth dongle was quite high. Finally, Wifi dongle was selected. It can provide a data rate of more than 10 MBps and range of approx 2 km.

2.4.2 Smoothing algorithms and Machine learning

As the image taken by the camera consisted of different types of noise such as salt noise, salt and pepper noise or Gaussian noise, various algorithms for noise reduction were studied and applied using OpenCV library. Also, as one of the main aim was to take images of a screen and then identify the digits being displayed on them. This required applying Machine Learning Algorithms such as Support Vector Machine(SVM), K-nearest neighbour and Artificial Neural Network. These were studied from [26] and OpenCV documentation.

Chapter 3

Overall Design

3.1 System Overview

The system can be visualised to have the following components with the interactions as shown:

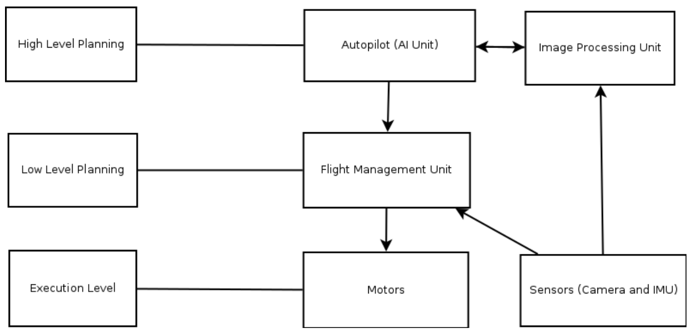


Figure 3.1: An overall view of the system

For this project, we have focussed on the execution level, the FMU, and the sensor integration into the FMU. We have also started development on the IP module.

3.2 Quad-rotor Dynamic Model

The quad-rotor has no moving parts and as a result has a simple mechanical design. However, its simplicity in design is challenged by its complicated dynamic system. The quad-rotor will be operating in two frames: the inertial frame and the body frame. The inertial frame is defined by the ground, and gravity points in the negative z direction. The body frame is defined by the orientation of the quad-rotor, with the rotor axes pointing in the positive z direction and the arms pointing in the x and y directions.

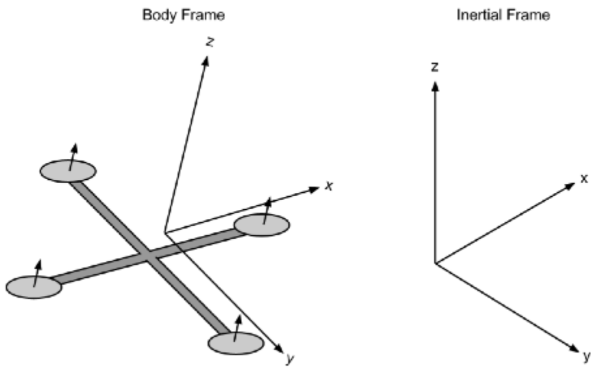


Figure 3.2: An overall view of the system

3.2.1 Kinematics

The position and velocity of the quad-rotor in the inertial frame is given by $x = (x, y, z)^T$ and $\dot{x} = (\dot{x}, \dot{y}, \dot{z})^T$, respectively. Similarly, the roll, pitch, and yaw angles in the body frame are given by $\theta = (\phi, \theta, \Psi)^T$, with corresponding angular velocities given by $\dot{\Theta} = (\dot{\phi}, \dot{\theta}, \dot{\Psi})^T$. Also, the angular velocity vector $\omega = \dot{\Theta}$ is a vector pointing along the axis of rotation. In order to convert these angular velocities into the angular velocity vector, the following relation is used:

$$\omega = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \dot{\theta} \tag{3.1}$$

where ω is the angular velocity vector in the body frame. The body and inertial frames are related by a rotation matrix R which goes from the body frame to the inertial frame. This matrix is derived by using the ZYZ Euler angle conventions and successively undoing the yaw, pitch, and roll.

$$R = \begin{bmatrix} c_\phi c_\Psi - c_\theta s_\phi s_\Psi & -c_\Psi s_\phi - c_\phi c_\theta s_\Psi & s_\theta s_\Psi \\ c_\theta c_\Psi s_\phi + c_\phi s_\Psi & c_\phi c_\theta c_\Psi - s_\phi s_\Psi & -c_\Psi s_\theta \\ s_\phi s_\theta & c_\phi s_\theta & c_\theta \end{bmatrix} \tag{3.2}$$

3.2.2 Equations of Motion

In the inertial frame, the acceleration of the quad-rotor is due to thrust, gravity, and drag. The thrust vector in the inertial frame can be obtained by using the rotation matrix R to map the thrust vector from the body frame to the inertial frame. Thus, the linear motion can be summarized as

$$\ddot{x} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + RT_B + F_D \tag{3.3}$$

where $\vec{x}$ is the position of the quad-rotor, g is the acceleration due to gravity, F_D is the drag force, and T_B is the thrust vector in the body frame.T_B and F_D are given by the following matrices:

$$T_B = \sum_{i=1}^{4} T_i = k \begin{bmatrix} 0 \\ 0 \\ \Sigma\omega_i^2 \end{bmatrix} \tag{3.4}$$

$$F_D = \begin{bmatrix} -k_d\dot{x} \\ -k_d\dot{y} \\ -k_d\dot{z} \end{bmatrix} \tag{3.5}$$

While it is convenient to have the linear equations of motion in the inertial frame, the rotational equations of motion are useful in the body frame, so that express rotations can be expressed about the center of the quad-rotor instead of about the inertial center. The rotational equations of motion are derived from Euler's equations for rigid body dynamics. Expressed in vector form, Euler's equations are written as

$$I\dot{\omega} + \omega \times (I\omega) = \tau \tag{3.6}$$

where ω is the angular velocity vector, I is the inertia matrix, and τ is a vector of external torques. This can be rewritten as

$$\dot{\omega} = \begin{bmatrix} \dot{\omega_x} \\ \dot{\omega_y} \\ \dot{\omega_z} \end{bmatrix} = I^{\text{-}1}(\tau - \omega \times (I\omega)) \tag{3.7}$$

We can model our quad-rotor as two thin uniform rods crossed at the origin with a point mass (motor) at the end of each. This results in a simple diagonal inertia matrix of the form.

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \tag{3.8}$$

Thus, the final result for the body frame rotational equations of motion are:

$$\dot{\omega} = \begin{bmatrix} \tau_\phi I_{xx}^{-1} \\ \tau_\theta I_{yy}^{-1} \\ \tau_\Phi I_{zz}^{-1} \end{bmatrix} - \begin{bmatrix} \frac{I_{\mathrm{yy}}-I_{\mathrm{zz}}}{I_{\mathrm{xx}}}\omega_y\omega_z \\ \frac{I_{\mathrm{zz}}-I_{\mathrm{xx}}}{I_{\mathrm{yy}}}\omega_x\omega_z \\ \frac{I_{\mathrm{xx}}-I_{\mathrm{yy}}}{I_{\mathrm{zz}}}\omega_x\omega_y \end{bmatrix} \tag{3.9}$$

3.3 Prototype

Initially, an exploratory prototype was developed that focussed on basic flight dynamics to gain a better understanding of the flight characteristics of the quad-rotor. The initial prototype implemented a basic flight model that dealt with only motor thrusts, yaw, pitch and roll as an open control system. The aim behind the exploratory prototype was to gain a better understanding of the components being used and to test the basic functioning of each of those components.

The main control equation on this flight model was

$$T_n = T_{in} + K_y\, Y_{in} + K_p\, P_{in} + K_r\, R_{in} \tag{3.10}$$

where, T_{in} is the mapped control input for thrust, Y_{in} is the mapped control input for yaw, K_y is the multiplier for yaw input, P_{in} is the mapped control input for pitch, K_p is the multiplier for pitch input, R_{in} is the mapped control input for roll and K_r is the multiplier for roll input.

This prototype served as the basic structure for our current iteration of the flight management unit.

3.4 Control System Design

We intended to implement a linear time invariant system for the quad-rotor platform that would be running on the Arduino microcontroller. The general equations of an LTI are:

$$\dot{x} = A\, x + B\, u \tag{3.11}$$

$$y = C\,x + D\,u \tag{3.12}$$

where, A matrix is a 12×12 matrix. The matrix entries are governed by the physical forces that govern the quad-rotor motion. The B matrix is a 12×4 matrix whose entries are governed by the actuators we have in our system. The C matrix is a 4×12 matrix whose entries are governed by what sensors we have in our system. The D matrix is a 4×4 order matrix. The x matrix is a 12×1 order matrix and it contains the states of the system. The u matrix is a 4×1 order matrix and it contains the inputs to the system. The y matrix is a 4×1 order matrix and it contains the outputs of the system.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.13}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ k_1 & k_2 & k_3 & k_4 \\ k_5 & k_6 & k_7 & k_8 \\ k_9 & k_{10} & k_{11} & k_{12} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

3.5 PID Design

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and has been universally accepted in industrial

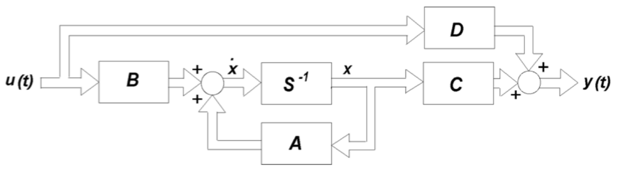


Figure 3.3: Control system statespace

control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner. As the name suggests, PID algorithm consists of three basic coefficients; proportional, integral and derivative which are varied to get optimal response. Closed loop systems are the primary application area of PID controllers. The PID controller can be described by the function:

$$u(t) = K_P\, error(t) + K_I \int error(t)\mathrm{d}t + K_D \frac{\mathrm{d}}{\mathrm{d}t}\, error(t). \tag{3.15}$$

The basic idea behind a PID controller is to read a sensor, then compute the desired actuator output by calculating proportional, integral, and derivative responses and summing those three components to compute the output. These responses are calculated based on an error term:

$$error = setpoint_{desired} - setpoint_{actual} \tag{3.16}$$

3.5.1 Proportional Response

The proportional component depends only on the difference between the set point and the process variable or the *error*. The proportional gain (K_P) determines the ratio of output response to the error signal. For instance, if the error term has a magnitude of 10, a proportional gain of 5 would produce a proportional response of 50. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If Kc is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control.

$$Proportional\, Term = K_P\, e(t) \tag{3.17}$$

3.5.2 Integral Response

The integral component sums the error term over time. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the Steady-State error to zero. Steady-State error is the final difference between the process variable and set point.

$$Integral\,Term = K_I \int e(t)\mathrm{d}t \tag{3.18}$$

3.5.3 Derivative Response

The derivative component causes the output to decrease if the process variable is increasing rapidly. The derivative response is proportional to the rate of change of the process variable. Increasing the derivative time (T_d) parameter will cause the control system to react more strongly to changes in the error term and will increase the speed of the overall control system response. Most practical control systems use very small derivative time (T_d), because the Derivative Response is highly sensitive to noise in the process variable signal. If the sensor feedback signal is noisy or if the control loop rate is too slow, the derivative response can make the control system unstable.

$$Derivative\,Term = K_D \frac{\mathrm{d}}{\mathrm{d}t} e(t) \tag{3.19}$$

3.6 PID Tuning

The process of setting the optimal gains for P, I and D to get an ideal response from a control system is called tuning. There are different methods of tuning of which the manual method and the Ziegler Nichols method will be used.

3.6.1 Manual Method

The gains of a PID controller can be obtained by trial and error. In this method, the I and D terms are set to zero and the proportional gain is increased until the output of the loop oscillates. As one increases the proportional gain, the system becomes faster, but care must be taken not make the system unstable. Once P has been set to obtain a desired fast response, the integral term is increased to stop the oscillations. The integral term reduces the steady state error, but increases overshoot. Some amount of overshoot is always necessary for a fast system so that it could respond to changes

Table 3.1: Ziegler Nichols Tuning Constants

Control Type	K_p	K_i	K_d
P	$0.50(K_u)$	-	-
PI	$0.45(K_u)$	$1.2(K_p)/P_u$	-
PID	$0.60(K_u)$	$2(K_p)/P_u$	$(K_p)(P_u)/8$

immediately. The integral term is tweaked to achieve a minimal steady state error. Once the P and I have been set to get the desired fast control system with minimal steady state error, the derivative term is increased until the loop is acceptably quick to its set point. Increasing derivative term decreases overshoot and yields higher gain with stability but would cause the system to be highly sensitive to noise.

3.6.2 Ziegler Nicholas Method

The Ziegler-Nichols method is another popular method of tuning a PID controller. It is very similar to the manual method wherein I and D are set to zero and P is increased until the loop starts to oscillate. Once oscillation starts, the critical gain Kc and the period of oscillations Pc are noted. The P, I and D are then adjusted as per the tabular column shown below.

3.7 DSP Design and Testing

We have considered filters in the time domain for signal smoothing i.e. the simple moving average (SMA), cumulative moving average (CMA), exponential moving average (EMA), the Savitzky-Golay filter, and the Ramer-Douglas-Peucker (RDP) algorithm.

3.7.1 Simple Moving Average

The SMA is the mean of the set of data points distributed uniformly on either side of a central value. The computed mean is then set as the control value in place of the received value at that time instant. The number of data points, also known as the window size can be varied.

3.7.2 Cumulative Moving Average

The CMA takes the mean of all the data that has arrived till that particular instant. It is given by the formula:

$$C_{k+1} = \frac{x_{k+1} + xC_k}{k+1} \tag{3.20}$$

$$C_k = \frac{\sum_{i=1}^{k} x_i}{k} \tag{3.21}$$

where C_k is the cumulative average of k data points and x_k is the value of k^{th} data point. C_0 is taken as zero. Similar to SMA, we again compute the CMA and set it as the control value, in place of the received value at that time instant.

3.7.3 Exponential Moving Average

The EMA is a type of weighted moving average with an exponential weighting function [6]. The weighting function uses a continuously decreasing exponential function. A constant factor, alpha, represents the degree of decrease in successive weights, and lies between 0 and 1. A smaller value of alpha takes more of the previous readings into account for calculating the current signal value. The EMA for a series S can be calculated as:

$$S_1 = C_1 \quad \text{for } \; t = 1$$

$$S_t = \alpha Y_t + (1-\alpha)S_{t-1} \quad \text{for } \; t > 1 \tag{3.22}$$

where Y_t is the data value at a time instant t. S_t is the value of the EMA at any time instant t.

3.7.4 Savitzky Golay Filter

The SG filter is a digital filter, which works on the method of fitting least squares, of a polynomial of a given order, to data points in a moving window. The polynomial is evaluated at the center of the moving window, and that value is the filtered value. The window is shifted over the entire data set and the central value of the window is calculated in each iteration of the algorithm. The degree of the polynomial used for least squares fitting and the moving window size can be varied and tested to obtain the most appropriate combination of parameters. The convolution coefficients for smoothing the signal can be obtained from the original paper of Savitzky and Golay [**?**].

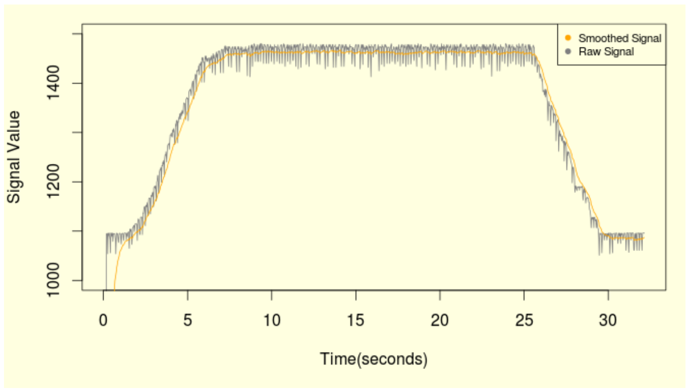


Figure 3.4: Plot representing input signal and smoothed output for EMA 0.10

3.7.5 Ramer Douglas Peucker Algorithm

The final algorithm implemented was RDP algorithm [5]. Given a set of curves, it attempts to reduce the number of points required to best approximate the curve. The algorithm uses the perpendicular distance of points from the curve to estimate which points need to be ignored. The ones at a distance greater than a fixed value (ϵ) are considered significant and remain part of the final smoothed output. It can be applied for smoothing noisy signals by filtering the data set to include fewer points.

3.7.6 Testing

Now that the algorithms were implemented, we executed the program on the Arduino board and observed the performance in real time. An open serial connection between the Arduino board and the workstation was used to collect data for a period of 30 seconds using the input signal as described above. Each record of the data set collected consisted of four parameters: time of call of smoothing function, raw value of input signal, time of return of smoothing function, and the smoothed signal value. We then analyzed the data and evaluated the performance of each algorithm.

3.8 Image Processing

The main aim of using image processing in this project is to identify the digits being displayed on a screen. The overall design of the image processing module is as follows:

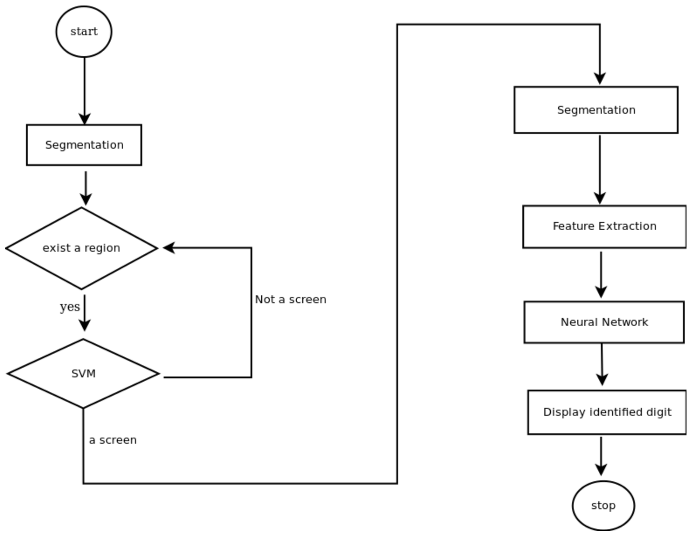


Figure 3.5: An overall view of the system

Firstly, we were required to capture the images from the camera and send it to the RPI. As the Open CV was installed on RPI, image processing was performed on RPI. Now we need to identify the digits in the captured images. This is basically a two step process:

1. Screen Localization: Extract the screen from the image.
2. Digits Identification: Extract the digits on the screen and then identify them.

For extracting the screen from the image i.e. the localization of the screen, SVM algorithm was used. Whereas for the second step, two algorithms k-nearest algorithm and artificial neural intelligence were applied and compared.

3.8.1 Screen Localization

Localizing the screen in the captured image itself is a two step process:

1. Segmentation: We apply different filters, morphological operations, contour algorithms, and validations to retrieve those parts of the image that could have a screen.
2. Classification: We apply a Support Vector Machine (SVM) classifier to each image patchour feature. Before creating our main application, we

train with two different classes: screen and non-screen images which acts as our training data.

Segmentation is a process of dividing the image into several segments and then finding and extracting the regions of our interest. One important feature of our image will be that it will be having lots of vertical lines. We can use this as out tool to segment the image. First, we convert our image to a gray scale image and then apply a sobel filter to extract vertical edges. Then we apply a threshold filter to obtain a binary image. After this we apply a close morphological operation in order to remove blank spaces between each vertical edge and connect all vertical lines that have a high number of edges. After this step we have regions in our image that could possibly have a screen. Then we find the external contours in the image and for each contour detected, extract the bounding rectangle of minimal area. We then make basic validations on its area and aspect ratio so as to classify them in a screen or a not screen region After we preprocess and segment all possible parts of an image, we now need to decide if each segment is (or is not) a screen. To do this, we will use a Support Vector Machine (SVM) algorithm. A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labelled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

One of the basic requirements of SVM is that it requires a lot of training data in order to classify the image. We trained our system with several screen images and several non-screen images. SVM classifier uses these images to train the system and saves the resultant information in an XML file. Finally it uses the information stored in an XML file in order to use them to detect screen in the test data.

3.8.2 Digits Identification

The second step aims to retrieve the characters on the screen with optical character recognition. For the detected screen, we proceed to find the position of digit in that image and then apply Artificial Neural Network (ANN) machine-learning algorithm to recognize the character.

ANN is more particularly a multi-layer perceptrons (MLP), the most commonly used type of neural networks. MLP consists of the input layer, output layer, and one or more hidden layers. Each layer of MLP includes one or more neurons directionally linked with the neurons from the previous and the next layer. Figure 3.6 represents a 3-layer perceptron with three inputs, two outputs, and the hidden layer including five neurons.

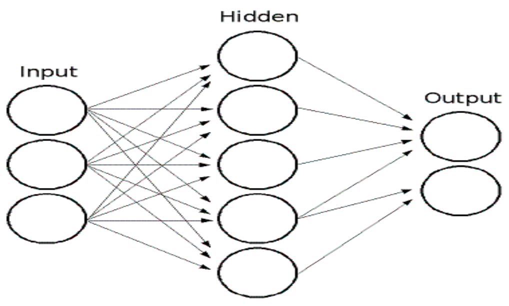


Figure 3.6: multi-layer perceptrons (MLP)

All the neurons in MLP are similar. Each of them has several input links (it takes the output values from several neurons in the previous layer as input) and several output links (it passes the response to several neurons in the next layer). The values retrieved from the previous layer are summed up with certain weights, individual for each neuron, plus the bias term. The sum is transformed using the activation function f that may be also different for different neurons.

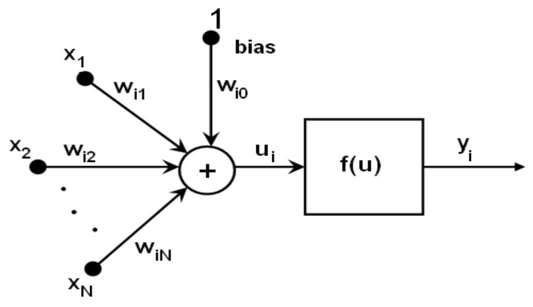


Figure 3.7: model of neural networks

The basic design of the second step is as shown in figure 3.8.

1. Dataset The data set used consists of a training set of 3050 samples (305 samples per digit).
2. Pre-Processing module The pre-processing module is responsible for converting our input data into a format, which is required by the neural network. The diagram 3.9 gives an overview of what happens to each image in the dataset.
   (a) Once the image is loaded, we apply GaussianBlur to remove noise and then apply thresholding to obtain a binary image.

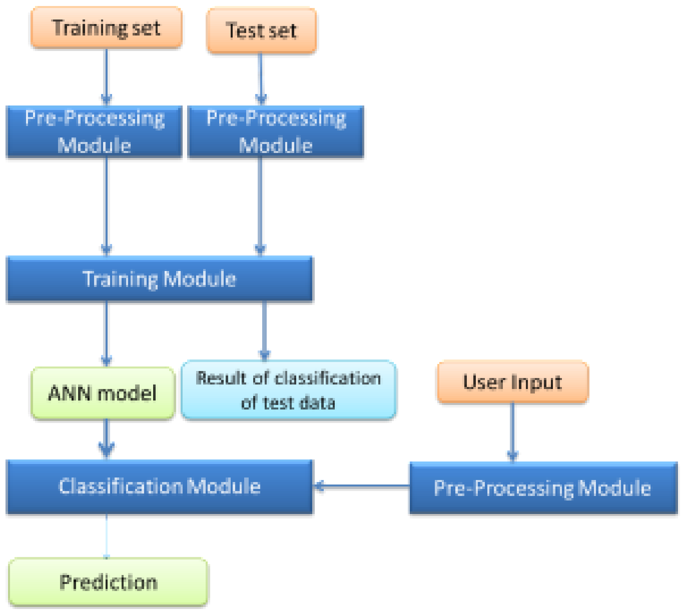


Figure 3.8: multi-layer perceptrons (basic design)

(b) Crop the image to eliminate the extra white space.

(c) Reduce the image dimension to 1616. (16 rows with 16 pixels each).

(d) Write the value of pixels into a file followed by the label, one image per row. For e.g. Consider the above image of 2. Each black pixel will be represented by 0 and white pixel by 1.So in the file we will write something like 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2

3. Training Module The training module is responsible for taking our transformed training set and generate the model of Neural Network.

4. Classification module This module will use the ANN model generated by the Training module to classify the user input.

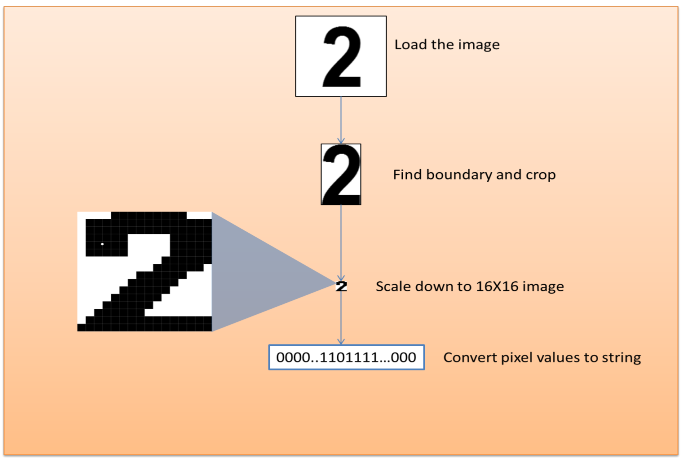


Figure 3.9: Pre-processing step

Chapter 4

Results and Performance

4.1 Control System Parameters

The control system parameters, namely matrices A, B, C and D as defined in (3.11) and (3.12) have partially been estimated. We are continuing work on establishing the remaining parameters. During the implementation of the control system on the Arduino platform, we realised that the limited memory of the platform did not provide sufficient resources for the complete implementation of the control system. Hence, we have implemented a partial control system, namely (3.12) in the current design. To solve this problem, we intend to move the control system to the Raspberry Pi, a board that we are already using onboard as the image processing unit. After some initial research we have decided on, ROS, an operating system designed specifically for robotics, to be our operating system of choice on the Raspberry Pi.

4.2 PID

The PID constant parameters for the quadrotor were found using the manual method. Tuning was also attempted using the Zieglar Nichols method. For tuning the system, we initially worked with an open loop system to ensure the safety of the quadrotor and all observers. We then allowed unrestricted motion on each of the three axes was using our testbed as the platform.

We have been able to attain optimal parameters for pitch and roll which due to symmetry in the system, can be assumed to be equivalent. K_P, K_I and K_D were found to be 0.80, and 0.05, for optimal response of the system. These parameters will further vary as we move forward and find out the remaining parameters.

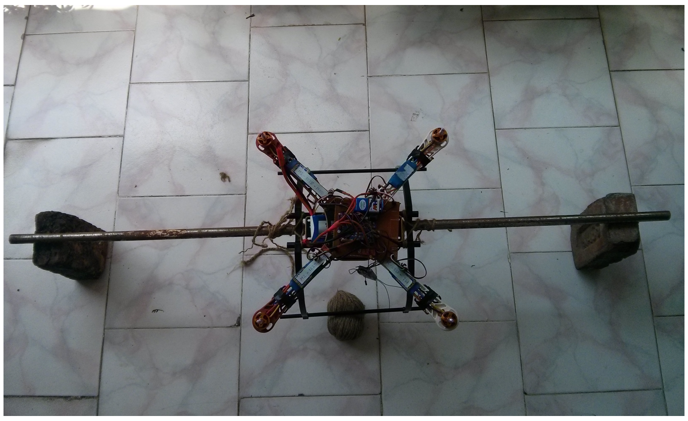


Figure 4.1: Topview of testbed

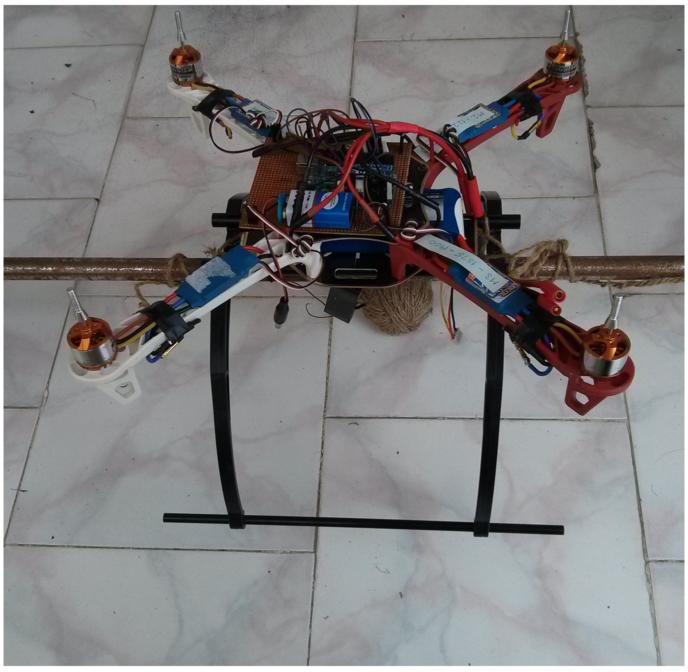


Figure 4.2: Sideview of testbed

4.3 DSP

As mentioned above, the data was collected in real-time from the MAV using a serial connection. The Arduino was refreshing control signal to the motors at a variable frequency depending on the algorithm being used. The data obtained was then preprocessed to remove any instances of invalid characters. An example of the input signal versus the smoothed output of EMA for alpha

0.10 is shown in figure 4.4.

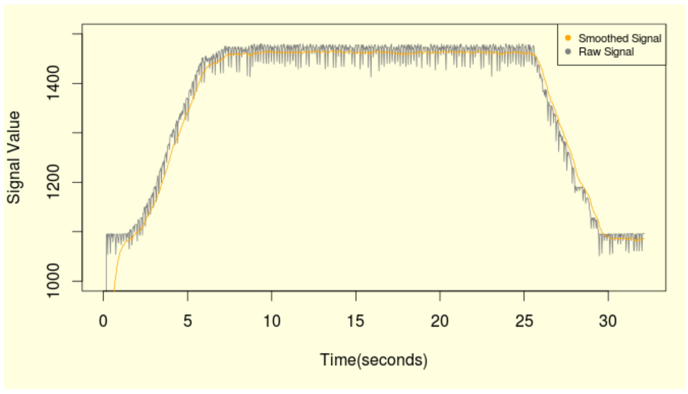
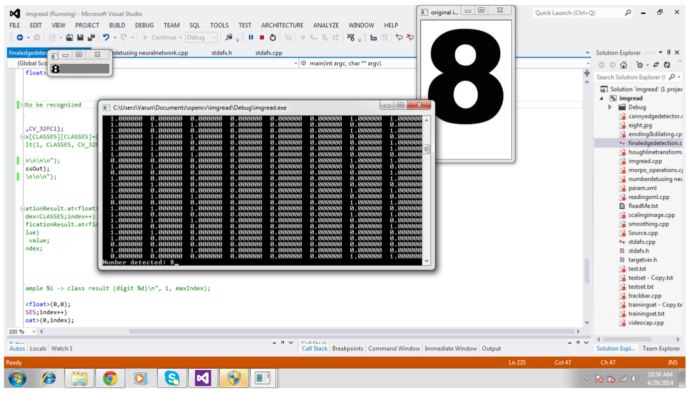


Figure 4.3: Control signal after DSP

The control input was processed in real time using various signal processing algorithms and their performance was evaluated. Exponential moving average filters were found to be the best performing algorithms with alpha parameters of 0.10 and 0.15. EMA was found to perform well on all metrics and came across as the best choice for application in a MAV.

4.4 IP

As mentioned in the previous section, Artificial Networks and K-nearest neighbour algorithm were applied to identify the digits being displayed on a screen. The following image shows the result displayed on the screen after applying Artificial Neural Networks.



```
C:\Users\Varun\Documents\opencv\imgread\Debug\imgread.exe
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  1.000000
 1.000000  1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  1.000000
 1.000000  1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
 1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
 1.000000  1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  1.000000
 1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
 1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  1.000000
 1.000000  1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  1.000000
 1.000000  1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  1.000000
 1.000000  1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000
 1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 1.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 1.000000  1.000000  1.000000  0.000000  0.000000  0.000000  0.000000  0.000000
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  1.000000  1.000000
Number detected: 8
```

Figure 4.4: Result of Artificial Neural Network

Artificial Neural Networks provided a better result i.e. less error percentage as compared to K-Nearest algorithm.

Chapter 5

Conclusion and Future Work

The project is presently in the tuning stages and we have attempted several tethered test flights. We have worked on and implemented various discrete modules that will be used in the complete flight control system. We constructed a testbed on which we are and will continue to tune the control system of the MAV. The PID constant parameters for pitch and roll have also been obtained. We have successfully implemented the algorithms for detecting the digits beig displayed on a screen. Alongside, a DSP module has been built which considers many algorithms to smooth the noisy control signal from the transmitter. This resulted in a significant improvement in the signal quality. We have resolved several issues encountered in this project to date, and we continue to work on outstanding issues. In the future, we will port the current control system to the Raspberry Pi platform, so as to be able implement a full-fledged control system. A significant portion of time will be invested into tuning the control system to perform optimally. On achieving that goal, we will move forward and start building an Aritificial Intelligence layer, on top of the FMU, which will replace the human pilot. This will require improving the image processing module. We will conduct more tethered test flights, and eventually move to untethered flights, followed by exclusive mission tasks. This project is expected to cover a lot more areas of robotics and artificial intelligence as we move forward.

Bibliography

[1] Ranjan, A. and Jetley P. *"MICAV,"* 2014, GitHub repository https://github.com/AsheeshR/MICAV

[2] Ranjan, A. and Jetley P. *"Microsmooth,"* 2014, GitHub repository https://github.com/AsheeshR/Microsmooth

[3] Ranjan, A. and Jetley P. *"Evaluation of Signal Smoothing Algorithms for a Quadrotor MAV,"* In Press, Internation Conference on Autonomous Robot Systems and Competitions.

[4] C. Galiski and R. bikowskib *"Some problems of micro air vehicles development,"* Bulletin of the Polish Academy of Sciences, Technical Sciences, Vol. 55, No. 1, 2007

[5] Douglas, D. H. and Peucker, T. K. (2011) *Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature*, in Classics in Cartography: Reflections on Influential Articles from Cartographica (ed M. Dodge), John Wiley & Sons, Ltd, Chichester, UK.

[6] Wilson, A. *"Event Triggered Analog Data Acquisition Using the Exponential Moving Average,"* Sensors Journal, IEEE , vol.PP, no.99, pp.1,1

[7] Koswatta, R.; Karmakar, N.C. *"Moving average filtering technique for signal processing in digital section of UWB chipless RFID reader,"* Microwave Conference Proceedings (APMC), 2010 Asia-Pacific , vol., no., pp.1304,1307, 7-10 Dec. 2010

[8] Gibiansky, A *"Quadcopter Dynamics, Simulation and Control,"* May 2013

[9] Mitra, S. *"Autonomous Quadcopter Docking System,"* 2013

[10] Ferguson, J.; Coulthard T.; Schastlivenko E. *"Autonomous Quadcopter,"* December 2012

[11] D. Hanafi, M. Qetkeaw, R. Ghazali, M. Than, W. Utomo and R. Omar, *"Simple GUI Wireless Controller of Quadcopter,"* Int'l J. of Communications, Network and System Sciences, Vol. 6 No. 1, 2013, pp. 52-59. doi: 10.4236/ijcns.2013.61006.

[12] Malgoza, D.; Mercedes E.; Smith S.; West J., *" Quadcopter Autonomus Surveillance Robot,"* 2010

[13] Magnussen, O.; Skonhaug K., *"Modelling, Design and Experimental Study for a Quadcopter System Construction,"* 2011

[14] Ruijie He; Prentice, S.; Roy, N., *"Planning in information space for a quadrotor helicopter in a GPS-denied environment,"* Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on , vol., no., pp.1814,1820, 19-23 May 2008

[15] Argentim, L.M.; Rezende, W.C.; Santos, P.E.; Aguiar, R.A., *"PID, LQR and LQR-PID on a quadcopter platform,"* Informatics, Electronics & Vision (ICIEV), 2013 International Conference on , vol., no., pp.1,6, 17-18 May 2013

[16] Gurdan, D.; Stumpf, J.; Achtelik, M.; Doth, K.-M.; Hirzinger, G.; Rus, D., *"Energy-efficient Autonomous Four-rotor Flying Robot Controlled at 1 kHz,"* Robotics and Automation, 2007 IEEE International Conference on , vol., no., pp.361,366, 10-14 April 2007

[17] Srikanth Govindarajan, Tarun Agarwal, Sai Kishan R., C. S. Suraj, G. Ramesh, and Veena Devi, *"Design of Multicopter Test Bench,"* International Journal of Modeling and Optimization vol. 3, no. 3, pp. 251-255, 2013.

[18] Bou-Ammar, H.; Voos, H.; Ertel, W., *"Controller design for quadrotor UAVs using reinforcement learning,"* Control Applications (CCA), 2010 IEEE International Conference on , vol., no., pp.2130,2135, 8-10 Sept. 2010

[19] Junior, Jos Claudio Vianna, et al. *"Stability Control of a Quad-rotor Using a PID Controller."* Brazilian Journal of Instrumentation and Control 1.1 (2013): 15-20.

[20] Ziegler, J.; Nichols, B.; *"Optimum Settings for Automatic Controllers"* Transactions of the ASME 64. pp. 759-768

[21] Palm, J.; Bradford A.; Nelson A.; *" Quadrotor UAV Project Final Report,"* 2010h" Quadrotor UAV Project Final Report," 2010

[22] Ahmadinejad, F.; Farad, H.; Ghidar, S.*"A Low-Cost Vision-Based Tracking System for Position Control of Quadrotor Robots,"* International Conference on Robotics and Mechatronics February 13-15, 2013

[23] *"AUVSI Student UAS Competition Team UAS DTU Journal Paper,"* 2011

[24] *"AUVSI Student UAS Competition Team UAS DTU Journal Paper,"* 2010

[25] *"AUVSI Student UAS Competition Team UAS-DTU Journal Paper,"* 2012

[26] Baggio, D.; Emami, S.; Escriv,D.; Ievgen, K.;Mahmood, N.;Saragih, J.;Shilkrot, R.*Mastering OpenCV with Practical Computer Vision Projects* First published: December 2012

[27] Bradski, G.; Kaehler, A.*"Learning OpenCV,"* September 2008: First Edition.